

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-097158  
(43)Date of publication of application : 08.04.1997

(51)Int.Cl. G06F 3/14  
G06F 3/14

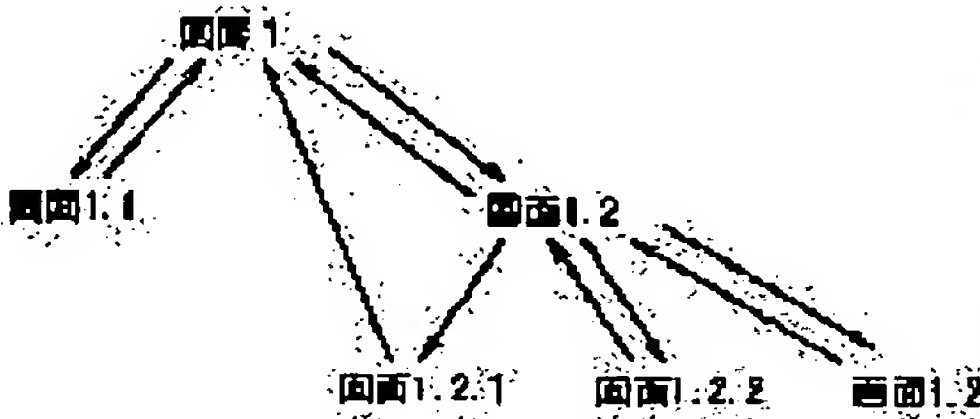
(21)Application number : 07-253716 (71)Applicant : SANYO ELECTRIC CO LTD  
(22)Date of filing : 29.09.1995 (72)Inventor : TAKAHASHI YUICHI

(54) SCREEN TRANSITION CONTROL METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To increase the response speed for screen switching by preliminarily generating the resource of a screen directly connected to the screen displayed at present.

SOLUTION: As a transition table shows the transition from a screen 1 to screen 1.1 and 1.2, a function realije is called to generate resources of these two screens. The screen 1.2 in the transition destination, five screens 1, 1.2, 1.1, 1.2, and 1.3 connected to it, and already generated resources are compared to determine the resource which is not included in these five screens, and a function destroy is called to abandon this resource, the resource of the screen 1.1 in this case. Thus, memory occupation of unnecessary resource is reduced. At the time of transition to a program which performs the processing related to the screen 1.2, a function map is only called to immediately display the picture 1.2 because the resource of the screen 1.2 is generated.



BEST AVAILABLE COPY

LEGAL STATUS

- [Date of request for examination]
- [Date of sending the examiner's decision of rejection]
- [Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
- [Date of final disposal for application]
- [Patent number]
- [Date of registration]
- [Number of appeal against examiner's decision of rejection]
- [Date of requesting appeal against examiner's decision of rejection]
- [Date of extinction of right]

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平9-97158

(43)公開日 平成9年(1997)4月8日

(51)Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 3/14	3 5 0		G 0 6 F 3/14	3 5 0 A
	3 4 0			3 4 0 A

審査請求 未請求 請求項の数3 O L (全 6 頁)

(21)出願番号 特願平7-253716

(22)出願日 平成7年(1995)9月29日

(71)出願人 000001889

三洋電機株式会社

大阪府守口市京阪本通2丁目5番5号

(72)発明者 高橋 祐一

大阪府守口市京阪本通2丁目5番5号 三

洋電機株式会社内

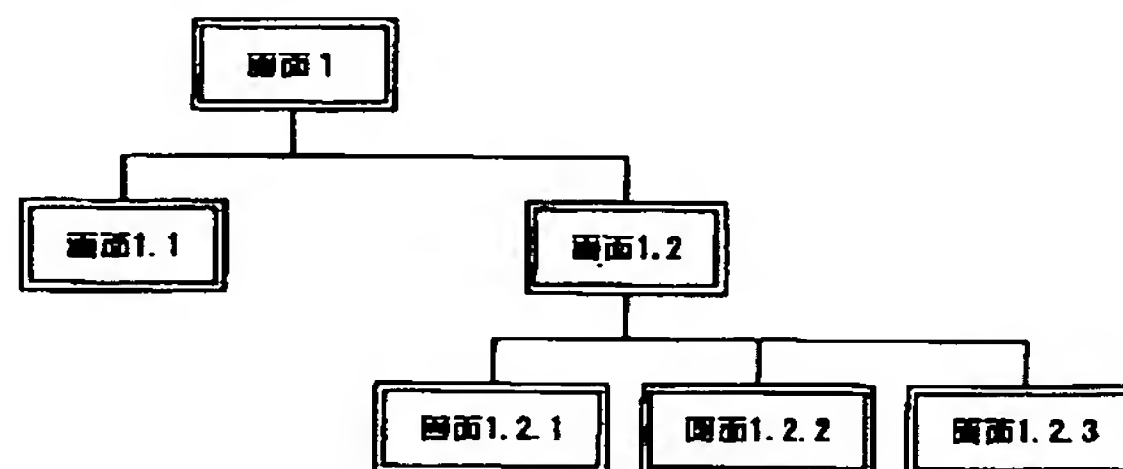
(74)代理人 弁理士 安富 耕二

(54)【発明の名称】 画面遷移の制御方法

(57)【要約】

【課題】 ウィンドウシステムにおいて画面のリソース生成に時間が掛かり、プログラム起動やI/Oの応答性が悪い。

【解決手段】 複数の画面を用意し、それらを切り替えてユーザと対話的に処理を進める情報処理装置であって、前記画面の表示に際して、該画面のリソースの生成と該リソースの表示とを分離して行いうる同装置において、前記画面相互の遷移方向を画面フローダイアグラムとして定義しておき、ひとつの画面を表示したとき、前記ダイアグラムによって当該画面と直接結び付けられた画面を決定し、次にそれらの画面についてのみあらかじめリソースを生成する。



## 【特許請求の範囲】

【請求項1】 複数の画面を用意し、それらを切り替えてユーザと対話的に処理を進める情報処理装置であって、前記画面の表示に際して、該画面のリソースの生成と該リソースの表示とを分離して行いうる同装置において、

前記画面相互の遷移方向を画面フローダイアグラムとして定義しておき、ひとつの画面を表示したとき、前記ダイアグラムによって当該画面と直接結び付けられた画面を決定し、次にそれらの又はその画面についてのみあらかじめリソースを生成するようになしたことを特徴とする画面遷移の制御方法。

【請求項2】 複数の画面を用意し、それらを切り替えてユーザと対話的に処理を進める情報処理装置であって、前記画面の表示に際して、該画面のリソースの生成と該リソースの表示とを分離して行いうる同装置において、

前記画面の遷移関係を表わす画面フローテーブルを定義しておき、ひとつの画面を表示したとき、前記画面フローテーブルを参照して当該画面に直接関係付けられた画面を決定し、次にそれらの又はその画面についてのみあらかじめリソースを生成するようになしたことを特徴とする画面遷移の制御方法。

【請求項3】 前記生成されたリソースのうち、画面の遷移に伴って遷移先の画面との直接の遷移関係がなくなった画面のリソースを、順次破棄することを特徴とする請求項1又は2記載の画面遷移の制御方法。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 多数の入力画面を用意し、それらを切り替えながらユーザが必要な情報を入力するようになした情報処理装置に関する。

## 【0002】

【従来の技術】 情報処理を行う場合、情報処理装置は、ユーザに対して必要な情報の入力を促すために、入力用画面あるいはダイアログボックスを表示し、ユーザからの入力を待つようにしている。また、対話的に処理を進めるために入力画面を複数用意し、ユーザが画面を切り替えながら、その画面、画面において要求されている情報を入力していくのが普通である。

【0003】 そして、この情報処理を行う画面は、一般に階層的な性質を持っている。即ち、図1は画面の移り替わりの関係を示したものであるが、例えば画面1からは、ユーザの指示により、画面1.2さらに画面1.2.1乃至1.2.3へと遷移することを示している。そして、画面1.2から画面1.1へ遷移したい場合には、ユーザは、一旦画面1に戻してから画面1.1に移る必要があることを示している。このように、情報処理装置においては、その画面の移り替わりに階層的な関係があり、それを図1のように表わすことができるのである。

【0004】 ところで、X-Wind owシステムなどの標準的なウィンドウシステムでは、ウィンドウが動作の基本単位となり、各アプリケーションプログラムは、物理画面上に重複して設定可能なウィンドウを使って、ユーザと対話するようになっている。また、特にユーザからの入力を受け付ける為に、ダイアログボックスと呼ぶ入力画面の一種をウィンドウ内に表示し、そのダイアログボックスを経由して、アプリケーションプログラムはユーザからの入力データを取り込むこともある。

10 【0005】 X-Wind owシステムにあっては、このダイアログボックスはインスタンスの集合として表わされる。インスタンスは視覚的に見えるグラフィカル・ユーザ・インターフェース（GUI）ひとつ、例えばボタンひとつに対してひとつ存在する構造体である。インスタンスは、それが表わすGUIのリソース、即ち、配置や大きさや色を定義していて、それぞれのインスタンスは、視覚的に最も下層となるインスタンスから順に、その階層関係に従って上下左右にポインタで接続されている。

20 【0006】 尚、ここでは、このインスタンスの集合体を便宜的にダイアログボックス又は画面と呼ぶことにする。そして、ある機能を持った一連のコードの中の関数が、ある画面のベースとなるインスタンスをパラメータとして呼び出されて、動作するのである。

【0007】 ここで、ひとつのダイアログボックスをウィンドウ内に表示するためには、先ず、関数realize()を使用して描画に必要なリソースを作り出さなければならない。

30 【0008】 関数realize() は、インスタンスのメンバを参照してカラーPixel値を決定し、指定されているフォントをロードし、サイズに合わせたウィンドウを作成する。そして、多くの場合、ビットマップにGUIの絵や文字を描画する。これは、ビットマップを基本にした方が表示速度が向上するからである。また、関数realize()は、ポインタで結ばれた同一レベルのインスタンス、さらに、下位のインスタンスが存在すれば、その全レコードに対して同様に処理してダイアログボックスのリソースを作り出す。

40 【0009】 ただし、関数realize() は表示のためのリソースを生成するだけであり、実際にそれを表示するには、関数map()を使用する。関数map()は同一レベル及び下位レベルのインスタンスを次々に表示し、サーバのイベントを受け付けることが出来るようにする。通常クライアント側には、送られてきたイベントに応じてウィンドウの内容、即ち画面あるいはダイアログボックスを表示するようなコードが記述されているので、結果としてウィンドウにダイアログボックスが表示される。

## 【0010】

50 【発明が解決しようとする課題】 図4は、アプリケーションプログラムの先頭部分における一般的な記述の例を

示す。初期設定として関数realize(Null,0) を呼び出すと、同関数は、当該ロードモジュール内に定義されているすべての画面あるいはダイアログボックスについて、前述した手順によりリソースを生成する。その際、生成したリソースはアプリケーションが管理するメモリに記憶するが、メモリが不足した場合、残りのリソースは補助記憶装置、例えばハードディスクに記憶する。

【0011】このように、アプリケーションプログラムの起動時にすべての画面のリソースを生成する場合、多数の画面を切り替えて処理を進めるようなプログラムでは、最初にリソース生成の為に時間が掛かってしまい、起動に要する時間が長くなってしまっていた。また、リソースが大量の場合、メモリ上に存在しないリソースが多くなる為、画面を切り替えた時にメモリとハードディスクとの間でスワッピングが発生する頻度が高くなり、その結果、画面の切り替えが遅くなる弊害も生じていた。

【0012】

【課題を解決するための手段】本発明は、複数の画面を用意し、それらを切り替えてユーザと対話的に処理を進める情報処理装置であって、前記画面の表示に際して、該画面のリソースの生成と該リソースの表示とを分離して行いうる同装置において、前記画面の遷移関係を表わす画面フローテーブルを定義しておき、ひとつの画面を表示したとき、前記フローテーブルを参照して当該画面に直接関係付けられた画面を決定し、次にそれらの又はその画面についてのみリソースを生成することにより、上記課題を解決するものである。

【0013】

【作用】本発明では、複数の画面を切り替えて処理を進める情報処理装置において、ある画面に切り替わったとき、その画面から直接遷移する画面を画面フローテーブルを参照して決定し、それらの画面についてのみ画面のリソースを生成する。そして、ユーザによって次の画面への切り替わりが指示されたとき、生成済みのリソースを使って画面を表示する。

【0014】

【実施例】図1は、実施例の情報処理装置が用意している画面とその階層関係を示す図である。実施例装置では先ず画面1を表示し、ユーザの指示あるいは入力に応じて、画面1.1又は画面1.2に表示を切り替えて新たな情報を提供するとともに、ユーザに対して次の指示あるいは入力を促すのである。

【0015】画面1.1からは、それより下層に結び付けられた画面が無いので、画面1に戻るしかない。一方、画面1.2には、その下層に画面1.2.1乃至1.2.3が存在し、ユーザの指示や入力に応じて、画面1.2から画面1.2.1、画面1.2.2又は画面1.2.3へと遷移することも出来るし、あるいは逆に、画面1に戻ることも出来る。

【0016】そして、画面1.2.1乃至1.2.3には、図から

わかるように、それより下層に画面が存在しないので、上層の画面つまり画面1.2に戻ることになるが、本実施例では特に、画面1.2.1からは直接画面1に戻るようになっている。

【0017】その画面遷移の方向をわかりやすく示したのが図2の画面フローダイアグラムである。画面1からは画面1.2を経由して画面1.2.1に達するが、そこからは画面1.2を経ずに画面1に遷移することがわかる。しかし、画面1.2.1はあくまで画面1.2の下層にあり、画面1からは直接画面1.2.1に遷移することが出来ないことが図からわかる。

【0018】図3は、図2に示す画面相互の遷移の関係を記述した遷移テーブル30である。同テーブル30において、左側に「現在の画面」として実施例装置に用意された6つの画面があり、それぞれの画面から図2において矢印で結び付けられた画面が右側の「遷移先の画面」にすべて定義されている。つまり、例えば、現在の画面が画面1ならば、そこから遷移する先の画面が画面1.1と画面1.2であることがわかるし、また、例えば、現在の画面が画面1.2.1であれば、遷移する先が画面1だけであることが、遷移テーブル30から読み取れるのである。

【0019】そして、本実施例では、アプリケーションプログラムを図5のように記述する。即ち、最初に表示する画面1に係わるプログラムでは、先ず、画面1を表示するために画面1を引数として関数realize("画面1",0) を呼び出し、画面1だけのリソースを生成する。そして、すぐに関数map("画面1",0)を呼び出してそのリソース、即ち、画面1を表示する。

【0020】通常はこの後、引き続いて画面1に係わる入出力処理を記述するが、本実施例では、その前に、現在の画面と結び付いた画面を遷移テーブル30を参照して決定し、その画面のリソースを生成する。この場合、画面1からは画面1.1と画面1.2に遷移することが遷移テーブル30からわかるので、その2つの画面のリソースを関数realize() を呼び出して生成する。

【0021】これは、計算機内部の処理に比較して人間の入力操作が極めて遅いので、画面1を表示してから実際の入力があるまで、計算機が何も仕事をしない時間が生じていたところ、その時間を利用して、次に表示されるであろう画面のリソースを、あらかじめ生成しておくものである。しかし、この段階ではリソースを生成するのみで、それを表示する関数map()を呼び出すことは無い。しかも、リソースを生成するのは、遷移テーブル30において現在の画面と直接結び付けられている画面だけである。

【0022】しかる後、画面1に係わるユーザからの指示や入力を受け付けることが出来るように、実際の処理を記述しておくのである。そして、例えば、ユーザの指示により画面1.2に遷移する場合は、画面1.2とは直接結び付きの無い現在生成済みのリソースを破棄してから、

遷移するようにしている。具体的には、遷移テーブル30を参照して、遷移先の画面1.2と、それに結び付けられた画面1及び画面1.2.1乃至1.2.3の5つの画面と、現在生成済みのリソースとを比較し、この5つの画面に含まれないリソースを決定して、そのリソース、この場合、画面1.1のリソースを関数destroy("画面1.1")を呼び出して破棄するようにしている。これにより、不要なリソースがメモリを占有することを減らし、ひいては、メモリとハードディスクとの間で発生するスワッピングの回数を最少にする。

【0023】その後、画面1.2に係わる処理を実行するプログラムに移行すると、図示はしていないが、そこでは、既に画面1.2のリソースが生成されているので、関数map("画面1.2",0)を呼び出すだけで直ちに画面1.2を表示できる。

【0024】そして、前述した処理と同様に、画面1.2に係わる入出力処理を記述する前に、現在の画面と結び付いた画面を遷移テーブル30を参照して決定し、その画面のリソースを生成する。この場合は、画面1.2からは画面1.2.1乃至1.2.3に遷移することがわかるので、その3つの画面のリソースを関数realize()を呼び出して生成するのである。そうしておいてから、画面1.2に固有の入出力処理を記述する。

【0025】以上の処理をフローチャートにまとめると図6のようになる。各々の画面に係わるプログラムは、最初に当該画面のリソースを表示するのであるが（ステップS603）、その前に当該画面のリソースが既に生成されているか否かを確認し（同S601）、無い場合にリソースを生成する（同S602）ようにするとよい。

【0026】次に、当該画面と結び付きのある画面を遷移テーブル30を参照して決定し（同S604）、それらの画面のリソースを生成する（同S605）。もちろんここでは、既に説明したようにリソースを生成するのみで、それを表示することはない。

【0027】その後、当該画面に係わる指示や入力を処理し（同S606）、次の画面に遷移するとき（同S607;Yes）、遷移テーブル30を参照して遷移先の画面において \*

\* 関係のなくなるリソースを決定する（同S608）。そして、不要になるリソースがあれば（同S609;Yes）、それらのリソースを破棄（同S610）してから、遷移先の画面に係わるプログラムに移行する（同S611）。

【0028】こうすることで、遷移先のプログラムでは、当該画面のリソースをいきなり表示することが出来るので、ユーザから見て画面切替の応答速度が速くなる。しかも、そのリソースは、画面表示してからユーザが入力応答するまでの待ち時間を利用して生成するので、ステップS606の入出力処理に影響を与えることもない。

【0029】

【発明の効果】本発明によれば、現在表示している画面に直接結び付いている画面のリソースをあらかじめ生成しておくので、画面の切り替えに際して応答時間が速くなる。

【0030】また、プログラムの起動時に全画面のリソースを一括して生成する従来の方法とは違い、直近に必要な画面のリソースのみ生成するので、プログラムの起動に要する時間を最少に出来る。しかも、不要なリソースがメモリを占有することがないので、メモリとハードディスクとの間の冗長なスワッピングを減少し、それによって、快適な応答性を実現出来る。

【図面の簡単な説明】

【図1】実施例装置が有する画面の階層関係を示す図である。

【図2】実施例装置が有する画面相互の遷移の方向を示す画面フローダイアグラムである。

【図3】画面相互の遷移の関係を記述した遷移テーブルである。

【図4】従来のプログラムの一般的な記述例を示す図である。

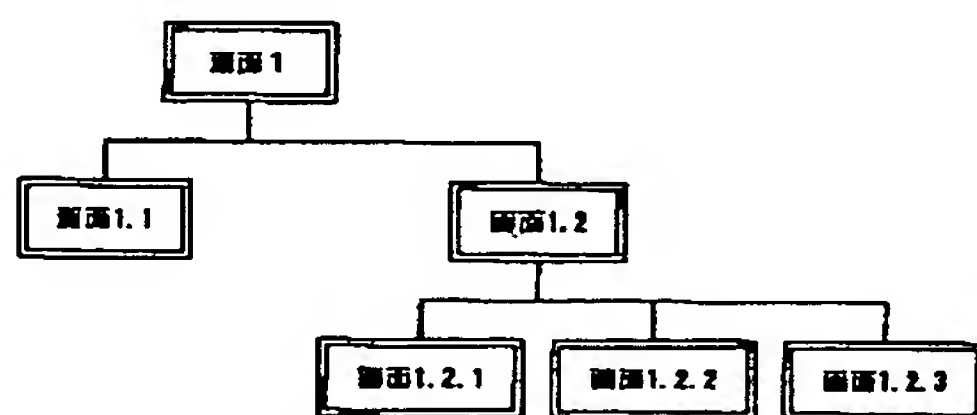
【図5】実施例プログラムの記述例を示す図である。

【図6】実施例の処理手順を示す図である。

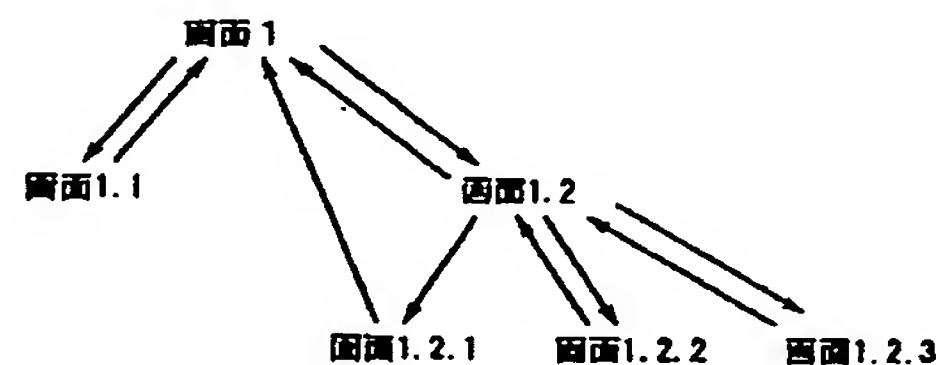
【符号の説明】

30 遷移テーブル

【図1】



【図2】



【図3】

現在の画面	遷移先の画面
画面1	画面1.1、画面1.2
画面1.1	画面1
画面1.2	画面1、画面1.2.1、画面1.2.2、画面1.2.3
画面1.2.1	画面1
画面1.2.2	画面1.2
画面1.2.3	画面1.2

30

【図4】

```

void main(int argc, char *argv[])
{
    /* 初期化処理 */
    .....

    /* 全画面のリソースを生成する */
    realize(tu11, 0);
    /* 全画面のリソースを類似的に表示する */
    map(tu11, 0);

    /* 画面1を表示する */
    map("画面1", 0);

    /* 画面1において入力を受け付ける */
    .....
}

```

【図5】

```

void main(int argc, char *argv[])
{
    /* 初期化処理 */
    .....

    /* 画面1のリソースを生成する */
    realize("画面1", 0);
    /* 画面1のリソースを表示する */
    map("画面1", 0);

    /* テーブルを参照して遷移先の画面を決定する */
    .....

    /* 遷移先画面のリソースを生成する */
    realize("画面1.1", 0);
    realize("画面1.2", 0);

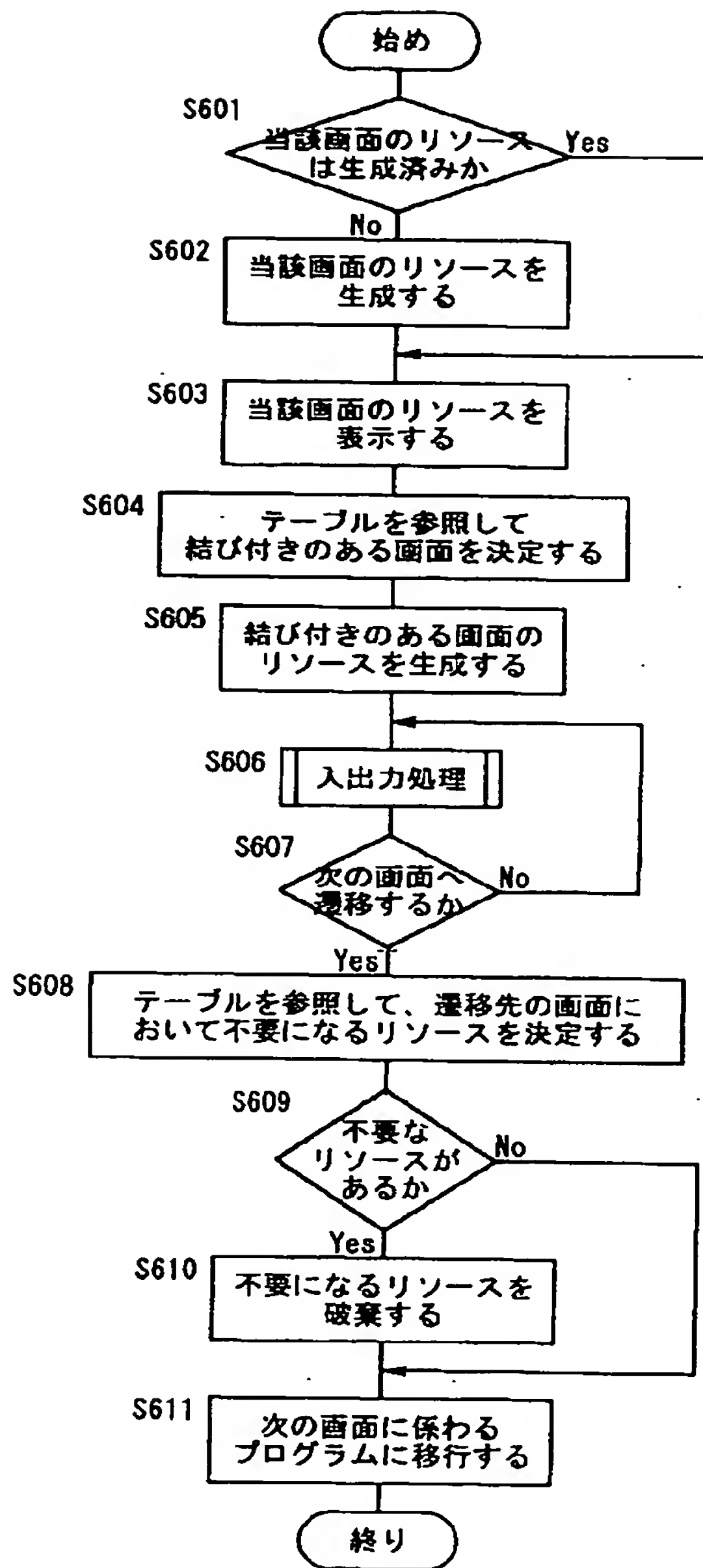
    /* 画面1において入力を受け付ける処理 */
    .....

    /* 画面1.2へ遷移する処理 : テーブルを参照して */
    /* 遷移先画面と関係の無いリソースを決定 */
    .....

    /* リソースを破壊する処理 */
    destroy("画面1.1", 0);
    /* 画面1.2へ遷移する */
    .....
}

```

【図6】



**This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record.**

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☒ **FADED TEXT OR DRAWING**
- ☒ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☒ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**